

Advanced XML Schema for NIEM

Modules Roadmap:

You Are Here

Anatomy of an XML Exchange

XML Conceptual Review

Basic XML Schema for NIEM



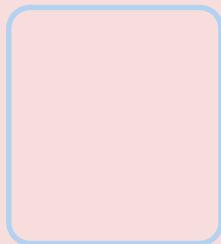
Advanced XML Schema for NIEM

Substitution Groups

Extension Schemas

Objectives Roadmap:

This module supports the following course objectives:



Define the physical components of an XML exchange.



Identify basic XML components that are used in the NIEM structure.



Write and/or extend an XML schema conformant to the NIEM Naming and Design Rules (NDR).

Module Objectives

- After completing this module, you should be able to:
 - ◆ Recognize the rationale behind namespaces.
 - ◆ Identify the physical realization of namespaces in XML schema.
 - ◆ Use schemas from another namespace.
 - ◆ Define the attributes and elements used in implementing namespaces.
 - ◆ Explain the concept of "import," "xmlns," and target namespace.
 - ◆ Explain the concept and usage of references.

Namespace Review

- A convention to uniquely identify an item in a distributed environment.
 - ◆ Needed when combining information from different sources.
 - ◆ Elements with the same name from different sources could have different meanings.
 - ◆ Abstract containers that provide context for items.
- The solution to naming conflicts.

Namespaces

What does “case” mean?

Court

any proceeding in a court of law whereby an individual seeks a legal remedy

Luggage

small or portable container for enclosing something, as for carrying or safekeeping

Food/Drink Containers

a box with its contents, case of soda

Construction

surrounding frame or framework, as of a door

Cards

the last card of a suit or denomination that remains after the other cards have been played

Why not define my own elements?

- Can re-use data defined by experts in that domain.
 - ◆ **Dublin Core** is an XML namespace for library classifications developed by librarians.
 - ◆ Why re-invent the wheel?

Namespace Construction

- Namespaces are declared by using the xmlns attribute

Standardized attribute that is applied to an element

URI
Uniform Resource Identifier

xmlns:nc="http://niem.gov/niem/niem-core/2.0"

Optional shortcut (or “Prefix”) to identify elements from a namespace in an XML document

URIs and Namespaces

- A URI is a globally unique identifier.
- Can be anything.
 - ◆ Commonly represented as a URL (Uniform Resource Locator)
 - i.e., web page identifiers, for example:
 - <http://www.w3.org/2001/XMLSchema>
 - <http://niem.gov/niem/niem-core/2.0>
 - May or may not be associated with a real web page.

Schema Namespace Declaration

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
    xmlns:abc="http://exampleNamespace/1.0"  
    xmlns:nc="http://niem.gov/niem/niem-core/2.0"  
    xmlns:j="http://niem.gov/niem/domains/jxdm/4.0"  
    xmlns:im="http://niem.gov/niem/domains/immigration/2.0">
```

Target Namespaces

- Specifies to which namespace the declared schema objects (elements, types, attributes) belong.
- Best Practice: One namespace per schema and vice versa.

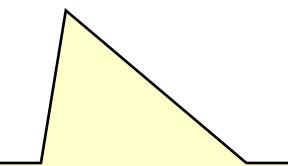
```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
    xmlns:abc="http://exampleNamespace/1.0"  
    xmlns:nc="http://niem.gov/niem/niem-core/2.0"  
    xmlns:j="http://niem.gov/niem/domains/jxdm/4.0"  
    xmlns:im="http://niem.gov/niem/domains/immigration/2.0"  
    targetNamespace="http://exampleNamespace/1.0">
```

Referencing Namespaces

- It is common that one namespace will leverage content from another namespace or namespaces.
- This is accomplished via an import statement.
- An import statement indicates where content from a specific namespace can be found.

Referencing Namespaces

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
    xmlns:abc="http://exampleNamespace/1.0"  
    xmlns:nc="http://niem.gov/niem/niem-core/2.0"  
    xmlns:j="http://niem.gov/niem/domains/jxdm/4.0"  
    xmlns:im="http://niem.gov/niem/domains/immigration/2.0"  
    targetNamespace="http://exampleNamespace/1.0">  
<xs:import namespace="http://niem.gov/niem/domains/jxdm/4.0"  
    schemaLocation="niem/domains/jxdm/4.0/jxdm.xsd"/>  
<xs:import  
    namespace="http://niem.gov/niem/domains/immigration/2.0"  
    schemaLocation="niem/domains/immigration/2.0/immigration.xsd"/>  
...  
...
```



Location specified relative to the directory this schema resides in.

XML Namespace Prefixes

- Associated to a namespace at time of declaration.
- Prefix is *not* the namespace.
- Can be thought of as a shortcut referring to the namespace URI.
- Qualifies objects to a specific namespace.

XML Namespace Prefixes

Prefix
associated
with this
schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
           xmlns:abc="http://exampleNamespace/1.0"  
           xmlns:nc="http://niem.gov/niem/niem-core/2.0"  
           targetNamespace=" http://exampleNamespace/1.0">  
  <xs:import namespace="http://niem.gov/niem/niem-core/2.0"  
            schemaLocation="niem/niem-core/2.0/niem-core.xsd"/>  
  <xs:complexType name="TestType">  
    <xs:sequence>  
      <xs:element ref="abc:Item"/>  
      <xs:element ref="nc:Item"/>  
    </xs:sequence>  
  </xs:complexType>  
  <xs:element name="Item" type="xs:string"/>  
</xs:schema>
```

Refers to Item
defined in this
schema

Refers to Item
defined in
niem-core

Exercise 4.1: Schemas

- Create two XML schema files:
 - ◆ Name one “base.xsd”
 - ◆ Name the other “extension.xsd”
- Use best practices mentioned in this module to create these schemas (details on next slide).
- When asked to define a new element/property it will always be of type “xs:string,” unless specified otherwise.
- When schemas are complete create two XML instances that would validate against extension.xsd

Exercise 4.1 Details

- base.xsd
 - ◆ Target namespace is <http://XML4NIEMTraining/base>
 - ◆ Namespace prefix for <http://XML4NIEMTraining/base> is “abc”
 - ◆ Create a global type named “VehicleType” with properties of “VehicleMake”, “VehicleModel”, and “VehicleColor”
 - ◆ Create a global element, "Vehicle" of type "VehicleType"
- extension.xsd
 - ◆ Target namespace is <http://XML4NIEMTraining/extension>
 - ◆ Namespace prefix for <http://XML4NIEMTraining/extension> is “def”
 - ◆ Import <http://XML4NIEMTraining/base>
 - ◆ <http://XML4NIEMTraining/base> has a prefix of “abc”
 - ◆ Create a global type named PersonNameType that contains globally defined given, middle and sur names
 - ◆ Create a global element, "PersonName" of type "PersonNameType"
 - ◆ Create a global type named “VehicleDriverAssociationsType”
 - ◆ Give this type element "Vehicle" from <http://XML4NIEMTraining/base> and element "PersonName" where the cardinality for each element is minimum of 1, maximum is unbounded.

Solution 4.1: Schemas *(1 of 3)*

```
<?xml version="1.0" encoding="UTF-8"?>
  <!--base.xsd -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:abc="http://XML4NIEMTraining/base"
  targetNamespace="http://XML4NIEMTraining/base">
  <xs:complexType name="VehicleType">
    <xs:sequence>
      <xs:element ref="abc:VehicleMake"/>
      <xs:element ref="abc:VehicleModel"/>
      <xs:element ref="abc:VehicleColor"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Vehicle" type="abc:VehicleType"/>
  <xs:element name="VehicleMake" type="xs:string"/>
  <xs:element name="VehicleModel" type="xs:string"/>
  <xs:element name="VehicleColor" type="xs:string"/>
</xs:schema>
```

Solution 4.1: Schemas (2 of 3)

```
<?xml version="1.0" encoding="UTF-8"?>
<!--extension.xsd→
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:abc="http://XML4NIEMTraining/base"
  xmlns:def="http://XML4NIEMTraining/extension" targetNamespace="http://XML4NIEMTraining/extension">
  <xs:import namespace="http://XML4NIEMTraining/base" schemaLocation="base.xsd"/>
  <xs:complexType name="PersonNameType">
    <xs:sequence>
      <xs:element ref="def:PersonGivenName"/>
      <xs:element ref="def:PersonMiddleName"/>
      <xs:element ref="def:PersonSurName"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="VehicleDriverAssociationsType">
    <xs:sequence>
      <xs:element ref="abc:Vehicle" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="def:PersonName" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="PersonName" type="def:PersonNameType"/>
  <xs:element name="PersonGivenName" type="xs:string"/>
  <xs:element name="PersonMiddleName" type="xs:string"/>
  <xs:element name="PersonSurName" type="xs:string"/>
  <xs:element name="VehicleDriverAssociations" type="def:VehicleDriverAssociationsType"/>
</xs:schema>
```

Solution 4.1: Schemas *(3 of 3)*

```
<?xml version="1.0" encoding="UTF-8"?>

<def:VehicleDriverAssociations xmlns:abc="http://XML4NIEMTraining/base"
xmlns:def="http://XML4NIEMTraining/extension">
    <abc:Vehicle>
        <abc:VehicleMake>String</abc:VehicleMake>
        <abc:VehicleModel>String</abc:VehicleModel>
        <abc:VehicleColor>String</abc:VehicleColor>
    </abc:Vehicle>
    <def:PersonName>
        <def:PersonGivenName>String</def:PersonGivenName>
        <def:PersonMiddleName>String</def:PersonMiddleName>
        <def:PersonSurName>String</def:PersonSurName>
    </def:PersonName>
</def:VehicleDriverAssociations>
```

Relationship Mechanisms

- XML has 2 basic mechanisms for associating data:
 - ◆ Hierarchical Inclusion of data
 - ◆ XML referencing

Hierarchical Inclusion

- Associated data is imbedded within another element:
 - ◆ Simple to interpret
 - ◆ Simple to process
 - ◆ Most people are familiar with it

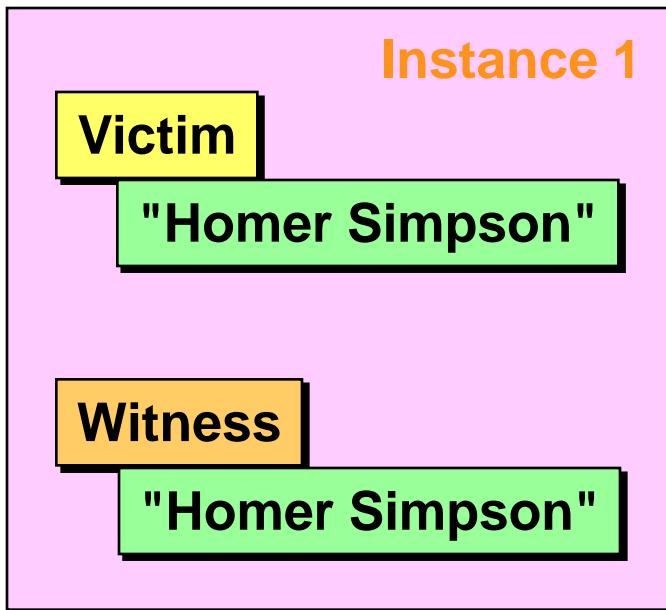


Example of Hierarchical Inclusion

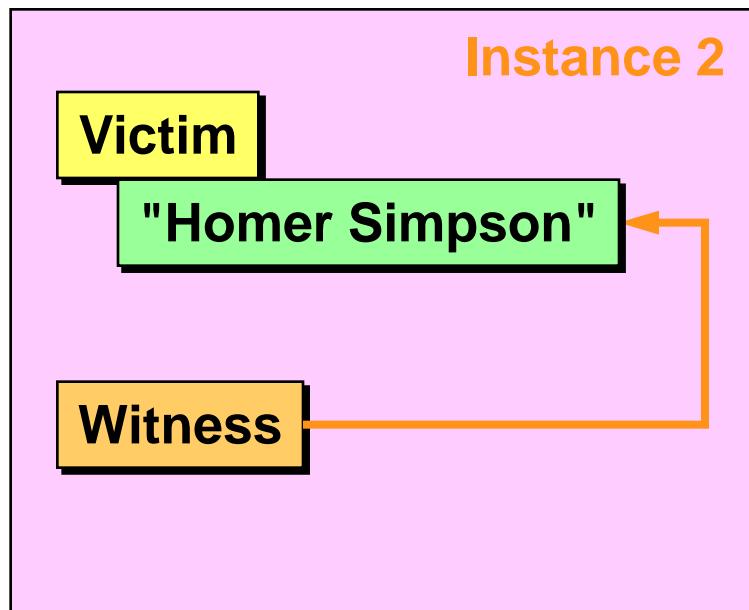
```
<Incident>
  <ActivityDescriptionText>
    Something happened
  </ActivityDescriptionText>
  <IncidentLocation>
    <LocationAddress>
      <AddressFullText>
        742 Evergreen Terrace
      </AddressFullText>
      <LocationCityName>Springfield</LocationCityName>
    </LocationAddress>
  </IncidentLocation>
</Incident>
```

XML Referencing

- Avoids redundancy
- Conveys equivalence
- Supported with XML Schema Types, "ID," "IDREF," and "IDREFS"



VS.



Schema Declarations

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:abc="http://exampleNamespace/1.0"
    targetNamespace="http://exampleNamespace/1.0">
    <xs:complexType name="PersonType">
        <xs:sequence>
            <xs:element name="GivenName" type="xs:string"/>
            <xs:element name="SurName" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="kin" type="xs:IDREFS" use="optional"/>
        <xs:attribute name="parent" type="xs:IDREF" use="optional" />
        <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:complexType>
    <xs:complexType name="PeopleType">
        <xs:sequence>
            <xs:element name="Person" type="abc:PersonType" minOccurs="1"
                maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="People" type="abc:PeopleType"/>
</xs:schema>
```

XML FOR NIEM: ADVANCED XML SCHEMA FOR NIEM

ID, IDREF, and
IDREFS must be
used as types for
declared attributes



XML Referencing Usage (1 of 2)

```
<People>
  <Person id="H1" kin="L1 B1" >
    <PersonGivenName>Homer</PersonGivenName>
    <PersonSurName>Simpson</PersonSurName>
  </Person>
  <Person id="B1" parent="H1" kin="H1 L1" >
    <PersonGivenName>Bart</PersonGivenName>
    <PersonSurName>Simpson</PersonSurName>
  </Person>
  <Person id="L1" parent="H1" kin="B1 H1" >
    <PersonGivenName>Lisa</PersonGivenName>
    <PersonSurName>Simpson</PersonSurName>
  </Person>
</People >
```

XML Referencing Usage (2 of 2)

```
<People>
  <Person id="H1" kin="L1 B1" >
    <PersonGivenName>Homer</PersonGivenName>
    <PersonSurName>Simpson</PersonSurName>
  </Person>
  <Person id="B1" parent="H1" kin="H1 L1" >
    <PersonGivenName>Bart</PersonGivenName>
    <PersonSurName>Simpson</PersonSurName>
  </Person>
  <Person id="L1" parent="H1" kin="B1 H1" >
    <PersonGivenName>Lisa</PersonGivenName>
    <PersonSurName>Simpson</PersonSurName>
  </Person>
</People >
```

The diagram illustrates XML referencing usage through three nested Person elements. The first Person element has an id of "H1" and a kin attribute with values "L1 B1". The second Person element has an id of "B1" and a parent attribute pointing to "H1", with a kin attribute of "H1 L1". The third Person element has an id of "L1" and a parent attribute pointing to "H1", with a kin attribute of "B1 H1". Green arrows highlight the 'kin' attribute of the first Person and the 'parent' attribute of the second Person. A pink arrow highlights the 'parent' attribute of the second Person and the 'id' attribute of the third Person, demonstrating how the 'parent' attribute of one node can reference the 'id' attribute of another node.

Exercise 4.2: Modify Schemas

- Using the schemas from Exercise 4.1, modify **base.xsd** as follows:
 - ◆ Add attribute id of type ID to VehicleType.
- Modify **extension.xsd** as follows:
 - ◆ Add attribute vehicleRef of type IDREF to PersonNameType.
- Generate an instance with 2 people and 2 vehicles using the attributes.

Solution 4.2: Modify Schemas *(1 of 3)*

```
<?xml version="1.0" encoding="UTF-8"?>
<!--base.xsd -->

<xsschema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:abc="http://XML4NIEMTraining/base"
  targetNamespace="http://XML4NIEMTraining/base">
  <xsc:complexType name="VehicleType">
    <xss:sequence>
      <xselement ref="abc:VehicleMake"/>
      <xselement ref="abc:VehicleModel"/>
      <xselement ref="abc:VehicleColor"/>
    </xss:sequence>
    <xsa:attribute name="id" type="xs:ID"/>
  </xsc:complexType>
  <xselement name="Vehicle" type="abc:VehicleType"/>
  <xselement name="VehicleMake" type="xs:string"/>
  <xselement name="VehicleModel" type="xs:string"/>
  <xselement name="VehicleColor" type="xs:string"/>
</xsschema>
```

Solution 4.2: Modify Schemas *(2 of 3)*

```
<?xml version="1.0" encoding="UTF-8"?>
<!--extension.xsd-->

<xsschema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:abc="http://XML4NIEMTraining/base" xmlns:def="http://XML4NIEMTraining/extension"
  targetNamespace="http://XML4NIEMTraining/extension">
  <xssimport namespace="http://XML4NIEMTraining/base" schemaLocation="base.xsd"/>
  <xsscomplexType name="PersonNameType">
    <xsssequence>
      <xsselement ref="def:PersonGivenName"/>
      <xsselement ref="def:PersonMiddleName"/>
      <xsselement ref="def:PersonSurName"/>
    </xsssequence>
  </xsscomplexType>
  <xsscomplexType name="VehicleDriverAssociationsType">
    <xsssequence>
      <xsselement ref="abc:Vehicle" minOccurs="1" maxOccurs="unbounded"/>
      <xsselement ref="def:PersonName" minOccurs="1" maxOccurs="unbounded"/>
    </xsssequence>
  </xsscomplexType>
  <xsselement name="PersonName" type="def:PersonNameType"/>
  <xsselement name="PersonGivenName" type="xs:string"/>
  <xsselement name="PersonMiddleName" type="xs:string"/>
  <xsselement name="PersonSurName" type="xs:string"/>
  <xsselement name="VehicleDriverAssociations" type="def:VehicleDriverAssociationsType"/>
</xsschema>
```

Solution 4.2: Modify Schemas *(3 of 3)*

```
<?xml version="1.0" encoding="UTF-8"?>
<def:VehicleDriverAssociations xmlns:abc="http://XML4NIEMTraining/base"
xmlns:def="http://XML4NIEMTraining/extension">
    <abc:Vehicle id="ID_1">
        <abc:VehicleMake>String</abc:VehicleMake>
        <abc:VehicleModel>String</abc:VehicleModel>
        <abc:VehicleColor>String</abc:VehicleColor>
    </abc:Vehicle>
    <abc:Vehicle id="ID_2">
        <abc:VehicleMake>String</abc:VehicleMake>
        <abc:VehicleModel>String</abc:VehicleModel>
        <abc:VehicleColor>String</abc:VehicleColor>
    </abc:Vehicle>
    <def:PersonName vehicleRef="ID_1">
        <def:PersonGivenName>String</def:PersonGivenName>
        <def:PersonMiddleName>String</def:PersonMiddleName>
        <def:PersonSurName>String</def:PersonSurName>
    </def:PersonName>
    <def:PersonName vehicleRef="ID_2">
        <def:PersonGivenName>String</def:PersonGivenName>
        <def:PersonMiddleName>String</def:PersonMiddleName>
        <def:PersonSurName>String</def:PersonSurName>
    </def:PersonName>
</def:VehicleDriverAssociations>
```

Module Summary

- After completing this module, you should be able to:
 - ◆ Recognize the rationale behind namespaces.
 - ◆ Identify the physical realization of namespaces in XML schema.
 - ◆ Use schemas from another namespace.
 - ◆ Define the attributes and elements used in implementing namespaces.
 - ◆ Explain the concept of "import," "xmlns," and target namespace.
 - ◆ Explain the concept and usage of references.

Creative Commons



Attribution-ShareAlike 2.0

You are free to

- Copy, distribute, display, and perform the work
- Make derivative works
- Make commercial use of the work



Attribution—You must give the original author credit



ShareAlike—If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one

Under the following conditions

- For any reuse or distribution, you must make clear to others the license terms of this work
- Any of these conditions can be waived, if you get permission from the copyright holder

Your fair use and other rights are in no way affected by the above

This is a human-readable summary of the [Legal Code \(the full license\)](#) and [Disclaimer](#)

This page is available in the following languages

[Català](#), [Deutsch](#), [English](#), [Castellano](#), [Suomeksi](#), [français](#), [hrvatski](#), [Italiano](#), [日本語](#), [Nederlands](#), [Português](#), and [中文\(繁\)](#)

[Learn how to distribute your work using this license](#)